الغرض من هذا البرنامج هو **تقديم نظام متكامل لإدارة مكتبة**، حيث يمكن من خلاله:

1. **إضافة الكتب** وتخزين معلوماتها الأساسية مثل العنوان والرقم التعريفي وعدد النسخ المتاحة.
2. **إدارة إعارة الكتب** للطلاب أو الأعضاء، مع تسجيل البيانات في قائمة انتظار (Queue) لضمان الإعارات بالترتيب الزمني.
3. **تسجيل إعادة الكتب** وتتبع آخر الكتب المعارة باستخدام **Stack**، بحيث يمكن استرجاع آخر عملية إعارة أو إرجاع بسرعة.
4. **إنتاج تقارير عن حالة المخزون** وعدد النسخ المتاحة لكل كتاب.
5. **حفظ واسترجاع البيانات من ملف خارجي** لضمان استمرار المعلومات بين جلسات التشغيل.

C#

```
using System;
using System.Collections.Generic;
using System.IO;

class LibrarySystem
{
    const string DATA_FILE = "library_books.txt";

    // Dictionary to store books (Book ID => Book)
    static Dictionary<string, Book> books = new
Dictionary<string, Book>();

    // Queue to manage loan requests (FIFO)
    static Queue<string> loanQueue = new Queue<string>();

    // Stack to track last returned books (LIFO)
    static Stack<string> returnStack = new Stack<string>();

    // Struct for book summary
    struct BookSummary
    {
        public string Title;
```

```
    public int AvailableCopies;
}

class Book
{
    public string Title;
    public int TotalCopies;
    public int AvailableCopies;
}

static void AddBook()
{
    Console.Write("Book ID: ");
    string id = Console.ReadLine();
    Console.Write("Book Title: ");
    string title = Console.ReadLine();
    Console.Write("Total Copies: ");
    int total = int.Parse(Console.ReadLine());

    books[id] = new Book { Title = title, TotalCopies = total,
AvailableCopies = total };
    Console.WriteLine("Book added successfully.");
}

static void LoanBook()
{
    Console.Write("Book ID to loan: ");
    string id = Console.ReadLine();

    if (!books.ContainsKey(id))
    {
        Console.WriteLine("Book not found.");
        return;
    }

    if (books[id].AvailableCopies <= 0)
    {
        Console.WriteLine("No copies available. Adding to loan
queue.");
```

```
            loanQueue.Enqueue(id);
            return;
        }

        books[id].AvailableCopies--;
        Console.WriteLine($"Book '{books[id].Title}' loaned
successfully.");
    }

    static void ReturnBook()
    {
        Console.Write("Book ID to return: ");
        string id = Console.ReadLine();

        if (!books.ContainsKey(id))
        {
            Console.WriteLine("Book not found.");
            return;
        }

        books[id].AvailableCopies++;
        returnStack.Push(id);
        Console.WriteLine($"Book '{books[id].Title}' returned
successfully.");

        // Process waiting loan queue
        if (loanQueue.Count > 0)
        {
            string queuedId = loanQueue.Peek();
            if (books[queuedId].AvailableCopies > 0)
            {
                loanQueue.Dequeue();
                books[queuedId].AvailableCopies--;
                Console.WriteLine($"Queued loan processed:
'{books[queuedId].Title}' loaned.");
            }
        }
    }
```

```
    static void GenerateReport()
    {
        Console.WriteLine("\nLibrary Inventory Report:");
        foreach (var b in books)
        {
            BookSummary summary = new BookSummary { Title =
b.Value.Title, AvailableCopies = b.Value.AvailableCopies };
            Console.WriteLine($"Book ID: {b.Key}, Title:
{summary.Title}, Available:
{summary.AvailableCopies}/{b.Value.TotalCopies}");
        }
    }

    static void ViewLastReturnedBook()
    {
        if (returnStack.Count == 0)
        {
            Console.WriteLine("No returned books yet.");
            return;
        }

        string lastId = returnStack.Peek();
        Console.WriteLine($"Last returned book:
{books[lastId].Title}");
    }

    static void Save()
    {
        using (StreamWriter sw = new StreamWriter(DATA_FILE))
        {
            foreach (var b in books)
            {
                sw.WriteLine($"{b.Key}|{b.Value.Title}|
{b.Value.TotalCopies}|{b.Value.AvailableCopies}");
            }
        }
    }

    static void Load()
```

```
    {
        if (!File.Exists(DATA_FILE)) return;

        foreach (var line in File.ReadAllLines(DATA_FILE))
        {
            var parts = line.Split('|');
            books[parts[0]] = new Book
            {
                Title = parts[1],
                TotalCopies = int.Parse(parts[2]),
                AvailableCopies = int.Parse(parts[3])
            };
        }
    }

    static void Main()
    {
        Load();
        while (true)
        {
            Console.WriteLine("\n1 Add Book  2 Loan Book  3 Return
Book  4 Inventory Report  5 View Last Returned  6 Exit");
            string choice = Console.ReadLine();

            switch (choice)
            {
                case "1": AddBook(); break;
                case "2": LoanBook(); break;
                case "3": ReturnBook(); break;
                case "4": GenerateReport(); break;
                case "5": ViewLastReturnedBook(); break;
                case "6": Save(); return;
                default: Console.WriteLine("Invalid option."); break;
            }
        }
    }
}
```

VB.NET

```
Imports System
Imports System.Collections.Generic
Imports System.IO

Module LibrarySystem

    Const DATA_FILE As String = "library_books_vb.txt"

    ' Dictionary to store books
    Dim books As New Dictionary(Of String, Book)()

    ' Queue for loan requests
    Dim loanQueue As New Queue(Of String)()

    ' Stack for last returned books
    Dim returnStack As New Stack(Of String)()

    ' Struct for book summary
    Structure BookSummary
        Public Title As String
        Public AvailableCopies As Integer
    End Structure

    Class Book
        Public Title As String
        Public TotalCopies As Integer
        Public AvailableCopies As Integer
    End Class

    Sub AddBook()
        Console.Write("Book ID: ")
        Dim id As String = Console.ReadLine()
        Console.Write("Book Title: ")
        Dim title As String = Console.ReadLine()
        Console.Write("Total Copies: ")
        Dim total As Integer = Integer.Parse(Console.ReadLine())
```

```
        books(id) = New Book With {.Title = title, .TotalCopies =
total, .AvailableCopies = total}
        Console.WriteLine("Book added successfully.")
    End Sub

    Sub LoanBook()
        Console.Write("Book ID to loan: ")
        Dim id As String = Console.ReadLine()

        If Not books.ContainsKey(id) Then
            Console.WriteLine("Book not found.")
            Return
        End If

        If books(id).AvailableCopies <= 0 Then
            Console.WriteLine("No copies available. Adding to loan
queue.")
            loanQueue.Enqueue(id)
            Return
        End If

        books(id).AvailableCopies -= 1
        Console.WriteLine($"Book '{books(id).Title}' loaned
successfully.")
    End Sub

    Sub ReturnBook()
        Console.Write("Book ID to return: ")
        Dim id As String = Console.ReadLine()

        If Not books.ContainsKey(id) Then
            Console.WriteLine("Book not found.")
            Return
        End If

        books(id).AvailableCopies += 1
        returnStack.Push(id)
```

```
        Console.WriteLine($"Book '{books(id).Title}' returned
successfully.")

        ' Process waiting loan queue
        If loanQueue.Count > 0 Then
            Dim queuedId As String = loanQueue.Peek()
            If books(queuedId).AvailableCopies > 0 Then
                loanQueue.Dequeue()
                books(queuedId).AvailableCopies -= 1
                Console.WriteLine($"Queued loan processed:
'{books(queuedId).Title}' loaned.")
            End If
        End If
    End Sub

    Sub GenerateReport()
        Console.WriteLine(vbCrLf & "Library Inventory Report:")
        For Each kvp In books
            Dim summary As New BookSummary With {
                .Title = kvp.Value.Title,
                .AvailableCopies = kvp.Value.AvailableCopies
            }
            Console.WriteLine($"Book ID: {kvp.Key}, Title:
{summary.Title}, Available:
{summary.AvailableCopies}/{kvp.Value.TotalCopies}")
        Next
    End Sub

    Sub ViewLastReturnedBook()
        If returnStack.Count = 0 Then
            Console.WriteLine("No returned books yet.")
            Return
        End If

        Dim lastId As String = returnStack.Peek()
        Console.WriteLine($"Last returned book:
{books(lastId).Title}")
    End Sub
```

```
    Sub Save()
        Using sw As New StreamWriter(DATA_FILE)
            For Each kvp In books
                sw.WriteLine($"{kvp.Key}|{kvp.Value.Title}|
{kvp.Value.TotalCopies}|{kvp.Value.AvailableCopies}")
            Next
        End Using
    End Sub

    Sub Load()
        If Not File.Exists(DATA_FILE) Then Return

        For Each line In File.ReadAllLines(DATA_FILE)
            Dim parts() As String = line.Split("|"c)
            books(parts(0)) = New Book With {
                .Title = parts(1),
                .TotalCopies = Integer.Parse(parts(2)),
                .AvailableCopies = Integer.Parse(parts(3))
            }
        Next
    End Sub

    Sub Main()
        Load()
        While True
            Console.WriteLine(vbCrLf & "1 Add Book" &
                              vbCrLf & "2 Loan Book" &
                              vbCrLf & "3 Return Book" &
                              vbCrLf & "4 Inventory Report" &
                              vbCrLf & "5 View Last Returned" &
                              vbCrLf & "6 Exit")

            Dim choice As String = Console.ReadLine()
            Select Case choice
                Case "1"
                    AddBook()
                Case "2"
```

```
                LoanBook()
            Case "3"
                ReturnBook()
            Case "4"
                GenerateReport()
            Case "5"
                ViewLastReturnedBook()
            Case "6"
                Save()
                Exit While
            Case Else
                Console.WriteLine("Invalid option.")
        End Select
    End While
End Sub
End Module
```

Java

```
import java.io.*;
import java.util.*;

public class LibrarySystem {

    private static final String DATA_FILE = "library_books_java.txt";

    // HashMap to store books
    private static Map<String, Book> books = new HashMap<>();

    // Queue for loan requests
    private static Queue<String> loanQueue = new
LinkedList<>();

    // Stack for last returned books
    private static Stack<String> returnStack = new Stack<>();

    // Class for book summary (struct equivalent)
    static class BookSummary {
        String title;
```

```
    int availableCopies;

    BookSummary(String title, int availableCopies) {
        this.title = title;
        this.availableCopies = availableCopies;
    }
}

static class Book {
    String title;
    int totalCopies;
    int availableCopies;
}

private static void addBook(Scanner sc) {
    System.out.print("Book ID: ");
    String id = sc.nextLine();
    System.out.print("Book Title: ");
    String title = sc.nextLine();
    System.out.print("Total Copies: ");
    int total = Integer.parseInt(sc.nextLine());

    Book b = new Book();
    b.title = title;
    b.totalCopies = total;
    b.availableCopies = total;
    books.put(id, b);

    System.out.println("Book added successfully.");
}

private static void loanBook(Scanner sc) {
    System.out.print("Book ID to loan: ");
    String id = sc.nextLine();

    if (!books.containsKey(id)) {
        System.out.println("Book not found.");
        return;
    }
```

```
        Book b = books.get(id);
        if (b.availableCopies <= 0) {
            System.out.println("No copies available. Adding to loan
queue.");
            loanQueue.add(id);
            return;
        }

        b.availableCopies--;
        System.out.println("Book '" + b.title + "' loaned
successfully.");
    }

    private static void returnBook(Scanner sc) {
        System.out.print("Book ID to return: ");
        String id = sc.nextLine();

        if (!books.containsKey(id)) {
            System.out.println("Book not found.");
            return;
        }

        Book b = books.get(id);
        b.availableCopies++;
        returnStack.push(id);
        System.out.println("Book '" + b.title + "' returned
successfully.");

        // Process loan queue
        if (!loanQueue.isEmpty()) {
            String queuedId = loanQueue.peek();
            Book queuedBook = books.get(queuedId);
            if (queuedBook.availableCopies > 0) {
                loanQueue.poll();
                queuedBook.availableCopies--;
                System.out.println("Queued loan processed: '" +
queuedBook.title + "' loaned.");
            }
```

```
        }
    }

    private static void generateReport() {
        System.out.println("\nLibrary Inventory Report:");
        for (Map.Entry<String, Book> entry : books.entrySet()) {
            Book b = entry.getValue();
            BookSummary summary = new BookSummary(b.title,
b.availableCopies);
            System.out.println("Book ID: " + entry.getKey() + ", Title:
" + summary.title +
                               ", Available: " + summary.availableCopies +
"/" + b.totalCopies);
        }
    }

    private static void viewLastReturnedBook() {
        if (returnStack.isEmpty()) {
            System.out.println("No returned books yet.");
            return;
        }

        String lastId = returnStack.peek();
        System.out.println("Last returned book: " +
books.get(lastId).title);
    }

    private static void save() {
        try (BufferedWriter bw = new BufferedWriter(new
FileWriter(DATA_FILE))) {
            for (Map.Entry<String, Book> entry : books.entrySet()) {
                Book b = entry.getValue();
                bw.write(entry.getKey() + "|" + b.title + "|" +
b.totalCopies + "|" + b.availableCopies);
                bw.newLine();
            }
        } catch (IOException e) {
            System.out.println("Error saving data: " +
e.getMessage());
```

```
        }
    }

    private static void load() {
        File file = new File(DATA_FILE);
        if (!file.exists()) return;

        try (BufferedReader br = new BufferedReader(new
FileReader(file))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split("\\|");
                Book b = new Book();
                b.title = parts[1];
                b.totalCopies = Integer.parseInt(parts[2]);
                b.availableCopies = Integer.parseInt(parts[3]);
                books.put(parts[0], b);
            }
        } catch (IOException e) {
            System.out.println("Error loading data: " +
e.getMessage());
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        load();

        while (true) {
            System.out.println("\n1 Add Book\n2 Loan Book\n3 Return
Book\n4 Inventory Report\n5 View Last Returned\n6 Exit");
            String choice = sc.nextLine();

            switch (choice) {
                case "1": addBook(sc); break;
                case "2": loanBook(sc); break;
                case "3": returnBook(sc); break;
                case "4": generateReport(); break;
```

```
                case "5": viewLastReturnedBook(); break;
                case "6": save(); return;
                default: System.out.println("Invalid option."); break;
            }
        }
    }
}
```

HTML + JavaScript Implementation

```
<!DOCTYPE html>
<html>
<head>
    <title>Library Book Loan & Inventory System</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        input, button { margin: 5px; }
        #report { margin-top: 20px; }
    </style>
</head>
<body>
    <h1>Library Book Loan & Inventory System</h1>

    <h2>Add Book</h2>
    <input id="bookId" placeholder="Book ID">
    <input id="bookTitle" placeholder="Book Title">
    <input id="bookCopies" placeholder="Total Copies"
type="number">
    <button onclick="addBook()">Add Book</button>

    <h2>Loan / Return</h2>
    <input id="loanId" placeholder="Book ID">
    <button onclick="loanBook()">Loan Book</button>
    <button onclick="returnBook()">Return Book</button>

    <h2>Reports</h2>
    <button onclick="generateReport()">Inventory
Report</button>
    <button onclick="viewLastReturned()">Last Returned
Book</button>

    <div id="report"></div>

<script>
let books = {}; // Map: bookId -> {title, totalCopies,
availableCopies}
let loanQueue = []; // Queue for loans
```

```
let returnStack = []; // Stack for last returned books

// Load data from localStorage
if (localStorage.getItem('books')) {
    books = JSON.parse(localStorage.getItem('books'));
    loanQueue = JSON.parse(localStorage.getItem('loanQueue'));
    returnStack = JSON.parse(localStorage.getItem('returnStack'));
}

function saveData() {
    localStorage.setItem('books', JSON.stringify(books));
    localStorage.setItem('loanQueue', JSON.stringify(loanQueue));
    localStorage.setItem('returnStack', JSON.stringify(returnStack));
}

function addBook() {
    const id = document.getElementById('bookId').value.trim();
    const title =
document.getElementById('bookTitle').value.trim();
    const total =
parseInt(document.getElementById('bookCopies').value);

    if (!id || !title || isNaN(total)) { alert("Please fill all fields.");
return; }

    books[id] = { title: title, totalCopies: total, availableCopies:
total };
    saveData();
    alert("Book added successfully.");
}

function loanBook() {
    const id = document.getElementById('loanId').value.trim();
    if (!books[id]) { alert("Book not found."); return; }

    if (books[id].availableCopies <= 0) {
        alert("No copies available. Added to loan queue.");
        loanQueue.push(id);
```

```
    } else {
        books[id].availableCopies--;
        alert(`Book '${books[id].title}' loaned successfully.`);
    }
    saveData();
}

function returnBook() {
    const id = document.getElementById('loanId').value.trim();
    if (!books[id]) { alert("Book not found."); return; }

    books[id].availableCopies++;
    returnStack.push(id);
    alert(`Book '${books[id].title}' returned successfully.`);

    // Process loan queue
    if (loanQueue.length > 0) {
        const queuedId = loanQueue[0];
        if (books[queuedId].availableCopies > 0) {
            loanQueue.shift();
            books[queuedId].availableCopies--;
            alert(`Queued loan processed: '${books[queuedId].title}'
loaned.`);
        }
    }
    saveData();
}

function generateReport() {
    let report = "<h3>Library Inventory Report:</h3><ul>";
    for (let id in books) {
        const b = books[id];
        report += `<li>Book ID: ${id}, Title: ${b.title}, Available: $
{b.availableCopies}/${b.totalCopies}</li>`;
    }
    report += "</ul>";
    document.getElementById('report').innerHTML = report;
}
```

```
function viewLastReturned() {
    if (returnStack.length === 0) {
        alert("No returned books yet.");
        return;
    }
    const lastId = returnStack[returnStack.length - 1];
    alert(`Last returned book: '${books[lastId].title}'`);
}
</script>
</body>
</html>
```